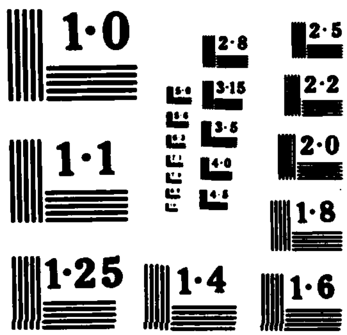
AD-A186 921   MAXIMUM LIKELIHOOD ESTIMATES FOR THE DISCRETE APPLICATION OF THE AMSAA GROWTH MODEL(U) ARMY MATERIEL SYSTEMS ANALYSIS ACTIVITY ABERDEEN PROVING GROU   1/1

UNCLASSIFIED   W P CLAY APR 87 AMSAA-TR-433   F/G 12/3   NL


1.0
2.8
2.5
3.15
2.2
3.5
1.1
4.0
2.0
4.5
1.8
1.25
1.4
1.6

(4)

AD

AMSAA

TECHNICAL REPORT NO. 433

MAXIMUM LIKELIHOOD ESTIMATES FOR THE DISCRETE APPLICATION OF THE AMSAA GROWTH MODEL

WILLIAM P. CLAY

APRIL 1987

U. S. ARMY MATERIEL SYSTEMS ANALYSIS ACTIVITY
ABERDEEN PROVING GROUND, MARYLAND

DISPOSITION

Destroy this report when no longer needed. Do not return it to the originator.

DISCLAIMER

The findings in this report are not to be construed as an official Department of the Army position unless so specified by other official documentation.

WARNING

Information and data contained in this document are based on the input available at the time of preparation. The results may be subject to change and should not be construed as representing the DARCOM position unless so specified.

TRADE NAMES

The use of trade names in this report does not constitute an official endorsement or approval of the use of such commercial hardware or software. The report may not be cited for purposes of advertisement.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER Technical Report No. 433	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER AD-A186921
4. TITLE (and Subtitle) Maximum Likelihood Estimates for the Discrete Application of the AMSAA Growth Model		5. TYPE OF REPORT & PERIOD COVERED Technical Report 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) William P. Clay		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS US Army Materiel Systems Analysis Activity Aberdeen Proving Ground, MD 21005-5071		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS US Army Materiel Command 5001 Eisenhower Avenue Alexandria, VA 22333		12. REPORT DATE April 1987 13. NUMBER OF PAGES 19
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Reliability Growth
Discrete Data
Maximum Likelihood Estimates

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report presents a computer program which will compute the maximum likelihood estimates of the parameters of the AMSAA reliability growth model when applied to discrete data. The solution algorithm, a modified bisection technique, is fully explained.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

ACKNOWLEDGEMENTS

The US Army Materiel Systems Analysis Activity (AMSAA) recognizes the following individual for contributing to this report.

Peer Reviewer: Joseph Wald, Air Warfare Division

The author wishes to thank the following:

Marian Brooks, Management & Technical Support Division (MTSD) - Typing Support.

Patricia Andrews, MTSD - Editing.

The next page is blank.

CONTENTS

The next page is blank.

MAXIMUM LIKELIHOOD ESTIMATES FOR THE
DISCRETE APPLICATION OF THE AMSAA GROWTH MODEL

1. INTRODUCTION

The U.S. Army Materiel Systems Analysis Activity (AMSAA) reliability growth model was developed under the leadership of Dr. Larry Crow during the mid to late 1970's. The resultant model was incorporated into Military Handbook 189, Reliability Growth Management, where it is fully explained. The model as developed by Dr. Crow is intended for use with continuous data. However, Dr. Crow later developed an as yet unpublished technique for applying the model to discrete data. Briefly, this technique assigns a sequence number to each trial of the test and uses these sequence numbers as the measures of "time" in the model. The growth parameters are estimated from the actual test data (the number of test intervals, the number of trials in each interval, and the number of failures in each interval) by the maximum likelihood method.

If the resultant likelihood equation had a true maximum, then any of several search algorithms could be applied to find it. However, as will be shown later, the likelihood equation can, in certain regions, be unbounded. The problem only becomes solvable if the maximum over a restricted region is sought. The purpose of this report is to present a solution to the problem and a computer program that implements that solution.

2. THE AMSAA RELIABILITY GROWTH MODEL AND DISCRETE DATA

Use of the AMSAA model assumes the test is divided into several "intervals." Each interval represents a portion of the test conducted on fixed configuration test items. One interval ends and another begins at any point in the test where significant changes are made in the test items in an attempt to change their reliability characteristics. Therefore, unless changes are applied periodically, it is inappropriate to use the model.

The model itself assumes that failures occur according to a nonhomogeneous Poisson process with a Weibull intensity function given as:

$$\rho(t) = \lambda\beta t^{\beta-1} \qquad [1]$$

where λ is the scale parameter and β is the growth or shape parameter. From [1] the expected number of failures on the interval $[T_{i-1}, T_i]$ is given by:

$$\lambda T_i^{\beta} - \lambda T_{i-1}^{\beta} \qquad [2]$$

and the average failure rate over the interval is:

$$\frac{\lambda (T_i^{\beta} - T_{i-1}^{\beta})}{T_i - T_{i-1}} \qquad [3]$$

To apply the model to discrete data, the sequence numbers of the last trial in each interval becomes that intervals "end time" and [3] becomes the average probability of failure for each trial in the ith interval. If K is defined as the number of intervals, $T_i - T_{i-1}$ as the number of trials in the ith interval and Mi as the number of failures in the ith interval, then a likelihood equation for the test can be constructed using [3] as the probability of failure for the ith interval and the binomial term determined from the test data for each interval. The likelihood function then is given by:

$$\ell = \prod_{i=1}^{K} \binom{T_i - T_{i-1}}{Mi} \left(\frac{\lambda T_i^{\beta} - \lambda T_{i-1}^{\beta}}{T_i - T_{i-1}} \right)^{M_i} \left(\frac{T_i - T_{i-1} - \lambda T_i^{\beta} + \lambda T_{i-1}^{\beta}}{T_i - T_{i-1}} \right)^{T_i - T_{i-1} - M_i} \quad [4]$$

The problem then becomes finding the values of λ and β which maximize [4]. To simplify the mathematics to follow, the following transforms are made:

Define $X_i = T_i^{\beta} - T_{i-1}^{\beta}$ [5a]
Define $N_i = T_i - T_{i-1}$ [5b]
Define $S_i = T_i - T_{i-1} - M_i$ [5c]
Define $F_i = M_i$ [5d]

Then [4] becomes

$$\ell = \prod_{i=1}^{K} \binom{N_i}{F_i} \left(\frac{\lambda X_i}{N_i} \right)^{F_i} \left(\frac{N_i - \lambda X_i}{N_i} \right)^{S_i} \quad [6]$$

3. MAXIMIZING THE LIKELIHOOD FUNCTION

Since the logarithm of F(X) maximizes at the same point as F(X), the values of λ and β which maximize [6] will also maximize.

$$\ln \ell = L = \sum_{i=1}^{K} (F_i) \ln(\lambda X_i) + \sum_{i=1}^{K} (S_i) \ln(N_i - \lambda X_i)$$

$$+ \sum_{i=1}^{K} \left[\ln \left(\binom{N_i}{F_i} \right) - \ln(N_i)^{F_i} - \ln(N_i)^{S_i} \right] \quad [7]$$

Note that the last term is a constant which can be eliminated without changing the values of λ and β which will maximize [7]. Therefore, the maximum likelihood estimates of λ and β ($\hat{\lambda}$, $\hat{\beta}$) can be found by maximizing:

$$L = \sum_{i=1}^{K} (F_i) \ln(\lambda X_i) + \sum_{i=1}^{K} (S_i) \ln(N_i - \lambda X_i) \quad [8]$$

To find λ and β from [8] would require that [8] be bounded and continuous. However, it is readily seen that for any fixed value of β, if λ increases without bound, the point is eventually reached where $N_1-\lambda X_1 = 0$. At this point L becomes discontinuous. In fact, by continuing this line of reasoning, it may be shown that for any fixed β there are K discontinuities corresponding to the points $N_1 = \lambda X_1$, $N_2 = \lambda X_2$, ..., $N_K = \lambda X_K$. To be able to solve [8] for $\hat{\lambda}$ and $\hat{\beta}$, then some restriction on [8] is needed that is not inconsistent with the intent of the solution. As it turns out, a real world limitation on [8] provides exactly such a restriction.

From [3], [5a], and [5b], it is noted that $\lambda X_i/N_i$ is an estimate of the probability of failure over the ith interval. Since it is a probability, it can be restricted to (0,1) without loss of generality for our problem (indeed, doing so enhances the solution). From this restriction, limits are imposed on λ namely,

$$0 < \lambda < \frac{N_i}{X_i} \qquad [9]$$

That is, λ must be such that it is less than N_i/X_i for each i. Coincidentally, this is precisely the restriction required to eliminate the singularities in [8]. So, for any fixed value of β, there is an upper limit, $\lambda max(\beta)$, imposed on λ. It may be shown that $\lambda max(\beta) = N_1/X_1$ for $\beta<1$ and $\lambda max(\beta) = N_K/X_K$ for $\beta>1$. [Note that $N_K/X_K = N_1/X_1 = 1 = \lambda max(1)$ at $\beta = 1$.] This yields a finite function, [8], over a restricted region defined by [9]. $\hat{\lambda}$ and $\hat{\beta}$ can now be sought.

Recalling that the X_i's are functions of β and defining X'_i to be the partial of X_i with respect to β, the first partials of [8] are given by:

$$\frac{\delta L}{\delta \lambda} = \sum_{i=1}^{K} \frac{F_i X_i}{\lambda X_i} - \sum_{i=1}^{K} \frac{S_i X_i}{N_i - \lambda X_i} \qquad [10]$$

$$\frac{\delta L}{\delta \beta} = \sum_{i=1}^{K} \frac{F_i \lambda X'_i}{\lambda X_i} - \sum_{i=1}^{K} \frac{S_i \lambda X'_i}{N_i - \lambda X_i} \qquad [11]$$

Continuing, the 2nd partials of [8] are given by:

$$\frac{\delta^2 L}{\delta \lambda^2} = -\sum \frac{F_i}{\lambda^2} - \sum \frac{S_i X_i^2}{(N_i - \lambda X_i)^2} \qquad [12]$$

$$\frac{\delta^2 L}{\delta \beta^2} = -\sum \frac{F_i T_i^{\beta} T_{i-1}^{\beta} (\ln T_i - \ln T_{i-1})}{X_i^2}$$

$$- \sum \frac{\lambda S_i X''_i}{(N - \lambda X_i)} - \sum \frac{\lambda^2 S_i (X'_i)^2}{(N - \lambda X_i)^2} \qquad [13]$$

$$\frac{\delta^2 L}{\delta\lambda\delta\beta} = -\sum \frac{\lambda S_i X_i \ X'_i}{(N-\lambda X_i)^2} - \sum \frac{S_i X'_i}{(N-\lambda X_i)} \qquad [14]$$

The two terms of [13] produced by taking the derivative of the first term of [11] have been combined to make it clear that the sign of the combined term is negative. In fact, since $T_i^{\ \beta}$, F_i, S_i, λ, X'_i, X''_i and $(\ln T_i - \ln T_{i-1})$ can all be shown to always be positive in the restricted region, it may be seen that [12], [13], and [14] are all negative everywhere in the restricted region. This indicates that [8] is concave down and there are no points of inflection over the entire region. Since [8] is also finite and continuous over the entire region, there are no local maxima and, if there is a point such that [10] and [11] are both zero, it is the global maximum. These results usually indicate the use of nonlinear programming algorithms to find the maximum. However, each of several techniques tried had problems involving either inordinate computational requirements or difficulty in dealing with the region boundry. Following these problems, the possibility of using a bracketing technique was investigated.

Consider the nature of [10] for some fixed β, say β_0. Since β is held constant, the X_i's are also constant.

If λ is set to zero, [10] becomes $+\infty$ because of the first term. If λ is set to $\lambda\max(\beta_0)$then either $N_1-\lambda X_1 = 0$ or $N_K-\lambda X_K = 0$ and [10] becomes $-\infty$. Now [12] shows that [10] monotonically decreases as λ increases and β is held constant. Therefore, the existence of some λ_0 such that [10] evaluated at (λ_0, β_0) equals zero is guaranteed. Since this argument can be made for any β_0, it indicates that there exists some function of β, call it $\lambda^*(\beta)$, such that [10] evaluated at $(\lambda^*(\beta_0), \beta_0)$ is equal to zero. Now, if $\lambda > \lambda^*(\beta_0)$ then [10] becomes negative (because it decreases monotonically) and L, consequently, decreases. If $\lambda < \lambda^*(\beta_0)$ then [10] becomes positive and L decreases. That is, λ^* (βo) defines the value of λ which maximizes [8] for $\beta = \beta_0$. The global maximum of [8] over the restricted region then must occur at one of these points. By algebraic manipulation of [10] and [5a], it can be shown that Figure 1 depicts the restricted region as viewed looking down on the λ, β plane.

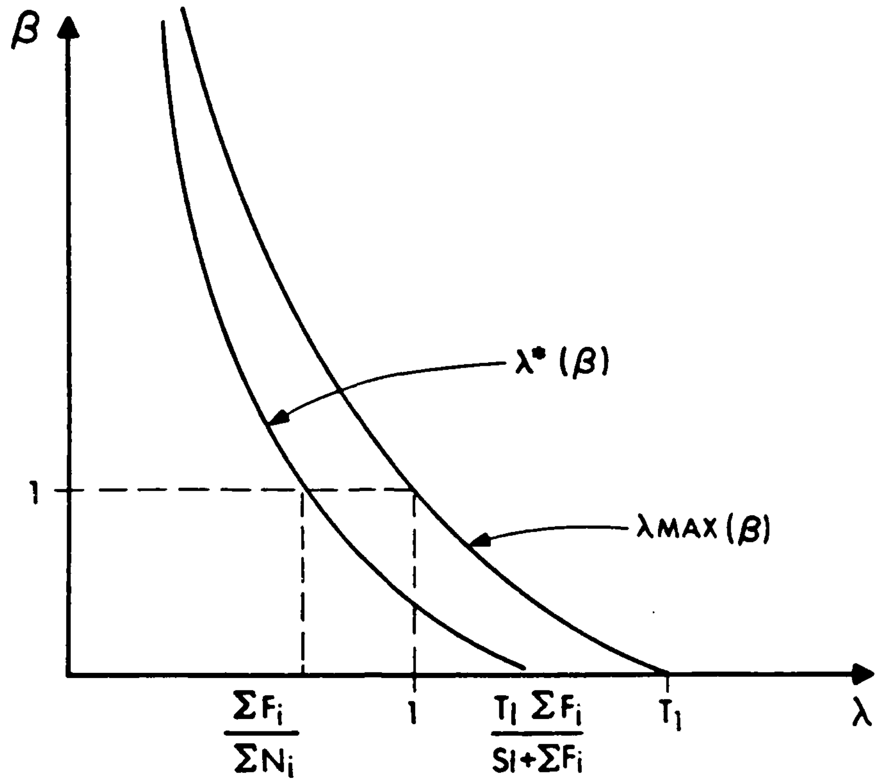


Figure. 1.

Figure 1 represents the general shape of the region. Specific values shown must occur. That is, $\lambda max(0)=T_1$,

$$\lambda^*(0) = \frac{T_1(\Sigma F_i)}{S_1+\Sigma F_i}, \quad \lambda max(1) = 1 \text{ and}$$

$$\lambda^*(1) = \frac{\Sigma F_i}{\Sigma N_i} \text{ always.}$$ The only possible deviation from Figure 1 for the specific points is that $\lambda^*(0)$ is less than 1 in a few special cases. For most legal datasets (that is, at least one failure and at least one success) $\lambda^*(0)$ is between 1 and T. The shape of $\lambda max(\beta)$ can be verified by simply plotting $\lambda max(\beta) = N_K/(T_K^{\beta}-T_{K-1}^{\beta})$ for $\beta<1$ and $\lambda max(\beta) = N_1/T_1^{\beta}$ for $\beta<1$. It has already been shown that $\lambda^*(\beta)$ must exist in the open interval $(0, \lambda max(\beta))$. Finally, the shape of $\lambda^*(\beta)$ can be inferred as follows. Select and β_0 and $\lambda^*(\beta_0)$. At this point [10] is zero. If β is increased to $\beta_0+\delta$ then [10] becomes negative, as indicated by [14] being negative (that is, an increase in β causes a decrease in [10]). Now, because [12] is negative everywhere, λ has to be decreased to bring [10] back to zero. Therefore, $\lambda^*(\beta+\delta) < \lambda^*(B)$ for positive δ.

All this indicates that $\lambda^*(\beta_0)$ can be found by a binary search technique. It is known that [10] is $+\infty$ at $\lambda=0$, $\beta=\beta_0$ and $-\infty$ at $\lambda = \lambda max(\beta_0)$, $\beta=\beta_0$. If [10] is then evaluated at $\lambda=\lambda max(\beta)/2$ and it is positive, then $\lambda^*(\beta_0)$ is known to be in the interval $(\lambda max(\beta_0)/2, \lambda max(\beta_0))$. Conversely, if [10] is negative $\lambda^*(\beta_0)$ is in the interval $(0, \lambda max(\beta_0)/2)$. This can be continued until either $\lambda^*(\beta_0)$ is found ([10] equals zero) or the width of the interval known to contain $\lambda^*(\beta_0)$ is less than the accuracy requirement in the solution.

This bracketing technique provides a means of locating $\lambda^*(\beta)$ for any given β. What remains is to develop a legitimate technique for searching along the function $\lambda^*(\beta)$ for the point that maximizes [8]. To this end, consider Figure 2.

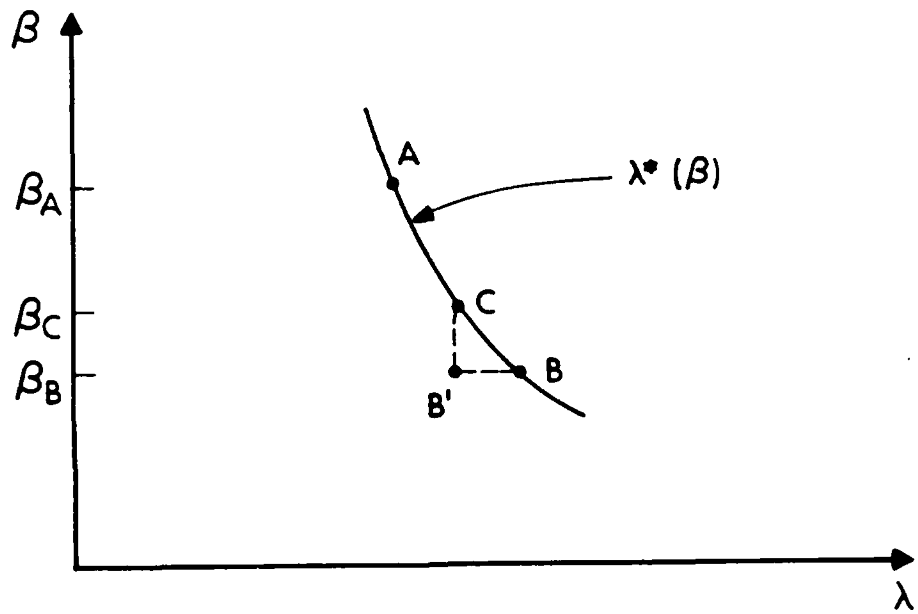


Figure 2.

Assume point B exists as a point on $\lambda^*(\beta)$ such that [11] is negative. That is, at B the slope of L with respect to λ is zero and the slope with respect to β is negative. Now assume that point A exists such that L evaluated at A (L_A) is greater than L_B. Since there are no points of inflection (and, therefore, no saddle points) in the restricted region, there must exist some point C between A and B, such that $L_C > L_B$, at any arbitrarily selected distance from B. Figure 1 depicts C as a point on $\lambda^*(\beta)$ but, for this argument, that is not required. Since C must exist at any radius arbitrarily close to B, it must exist at a radius such that point B' is arbitrarily close to B. Since [13] is continuous on the region a B' can always be selected so that [11] is still negative. Now $L_{B'} < L_B$ by definition of $\lambda^*(\beta)$. Also, since [14] is negative, increasing β from β_B to β_C causes L to decrease (since the slope of L with respect to β is negative). Therefore, $L_C < L_{B'} < L_B$. Since this violates the requirement that $L_C > L_B$ the initial assumption, that a point can exist at $\beta_A > \beta_B$ when [11] is negative and $L_A > L_B$ leads to a contradiction. Using a similar argument, it can be shown that if [11] is positive at some point $(\lambda^*(\beta_B), \beta_B)$ then there cannot exist any point $(\lambda^*(\beta_A), \beta_A)$ where $\beta_B > \beta_A$ and $L_A > L_B$.

As an end result, if [11] evaluated at some $(\lambda^*(\beta_0), \beta_0)$ is negative, then the global maximum will occur at some $\beta < \beta_0$. Conversely, if [11] evaluated at $(\lambda^*(\beta_0), \beta_0)$ is positive then any global maximum that exists will occur at some $\beta > \beta_0$. Note that there is no guarantee that a point exists such that [10] and [11] are both zero. It has not been proved that [11] cannot be asymptotic to zero as β increases without bound. In this case, however, there is a realistic limit imposed on β during the search. This limit is discussed in the next section.

This completes the proof that, theoretically, $\hat{\lambda}$, $\hat{\beta}$ can be found by a two dimensional binary search technique. The next section of this paper deals with implementing this theoretical approach on a digital computer.

4. FINDING $\hat{\lambda}$ AND $\hat{\beta}$ ON A DIGITAL COMPUTER

The previous section showed that, in theory, [8] has a maximum which can be located using a two dimensional search technique. This theory in its pure form, however, depends on being able to locate for any β a corresponding λ such that [10] is negative, but finite, and another such that [10] is positive, but finite. It also depends on being able to find a value of β such that [11] evaluated at $(\lambda^*(\beta), \beta)$ is negative. The existence of the first two points was proved in the previous section. It is possible, however, that while the points exist it may not be possible to compute them on a digital computer. It is possible to construct a problem such that for some β $\lambda^*(\beta)$ is so close to $\lambda max(\beta)$ as to be beyond the accuracy of the computer to discriminate between them. The theory, for instance, is satisfied by $\lambda max(\beta) = \lambda_0$ and $\lambda^*(\beta) = \lambda_0 - 2^{-50}$. (Assume λ_0 to be on the order of 1.) On a digital computer with, say, 23 bits for the mantissa of a floating point variable, the actual value computed for $\lambda max(\beta)$ may be as small $\lambda_0 - 2^{-24}$, which is less than $\lambda_0 - 2^{-50}$. That is, because of truncation error involved in the calculation of $\lambda max(\beta)$, it is possible that [10] evaluated at $(\lambda max(\beta), \beta)$ would be positive rather than negative. This result indicates that $\lambda^*(\beta)$ is too close to $\lambda max(\beta)$ for the computer to discriminate between the two.

Similarly, it may be that $\lambda^*(\beta)$ is so close to zero so as to be indistinguishable from zero by a computer. Based on experience with actual test cases, it is most likely that $\lambda^*(\beta)$ being computationally equal to zero will be encountered when the actual solution for $\hat{\beta}$ is very large (say $\hat{\beta} > 3$). This will cause $\lambda max(\beta)$, and consequently $\lambda^*(\beta)$, to pass very close to the β axis.

Finally, the existance of a point on $\lambda^*(\beta)$ such that [11] is negative has not been proved.

Fortunately, all of these situations have realistic work arounds. In the first case, the solution is to set $\lambda^*(\beta) = \lambda max(\beta)$ as computed and continue the algorithm. Because [8] is so very well behaved in the restricted region, small errors in the location of any particular parameter estimate has a correspondingly small effect on the solution. Experience with actual data indicates that truncation error in calculating $\lambda max(\beta)$ is enough to prevent [10] from evaluating to $-\infty$. In the second and third cases, an unsolvable problem is probably trivial. In several years of experience with the program outlined later there has never been a case where $\lambda^*(\beta)$

became computationally zero. Most of these runs have been made with a cutoff of 2 for β . That is, if [11] was positive at $(\lambda^*(2), 2)$ the program was stopped and a message that $\hat{\beta}$ was greater than 2 was printed. In other words, in every case ever run $\lambda^*(2)$ was non-zero. In real world terms, if $\beta > 2$ then the actual value really does not matter. If growth is occurring, then $\beta < 1$. For $\hat{\beta} > 1$ the reliability of the item is actually degrading as "improvements" are made. In short, determining that $\hat{\beta} > 2$ should be a sufficient result without knowing the actual value of $\hat{\beta}$.

5. A COMPUTER PROGRAM FOR COMPUTING $\hat{\lambda}$ AND $\hat{\beta}$

The appendix contains a computer program to compute λ and β. The program is written in FORTRAN 77 and was developed on a VAX 11/780 under the VMS operating system.

Main Program - The main program is provided for convenience and provides no part of the solution. It may be freely replaced with any other program which sets up the vectors N and F, provides for vectors X and DX and then calls SUBROUTINE MLHE.

SUBROUTINE MLHE - This is the routine that drives all others involved in the solution. Statements 30 to 35 initialize variables LAMBDA, BETA, EPS, TF, and TN.

Statements 36 to 47 do some error checking on the input vectors, N and F. An error condition (MSG≠0) is returned if 1) any interval has less than one trial, 2) any interval has less than zero failures, 3) any interval has more failures than trials, 4) all trials were failures, or 5) no trials were failures. Any of these conditions will cause the computation of λ and β to not take place.

Statements 48 - 53 set BETA to 2 (the artificial upper limit discussed in the last section) and computes the corresponding λ^* as well as the value of [11] by calling SUBROUTINE GETLAM. If the value of [11] (variable DB) is positive it indicates that $\hat{\beta} > 2$. In this case, the routine is terminated with MSG = 3, indicating that $\hat{\beta}$ is greater than 2.

Statements 54 - 69 perform the binary search for $\hat{\beta}$. Beginning with BH=2 and BL=zero. The value of DB is determined at (BH+BL)/2. If DB is greater than zero, BL is set to the average. If DB is less than zero, BH is set to the average. The loop is then executed again using the new BH and BL. The loop is terminated on either of two conditions: 1) The difference between BH and BL is less than the accuracy requirement (EPS), or 2) The number of trips through the loop exceeds 100. The second condition is necessary to prevent infinite loops under certain conditions. If too much accuracy (variable NSIG) is requested, it is possible to get into the situation where BH-BL computes to a value greater than EPS, but (BH+BL)/2 computes to BH or BL. This situation occurred on a VAX 11/780 under VMS FORTRAN when NSIG was set to 6.

Statements 70 - 71 take the average of the last values of BH and BL and call GETLAM to compute LAMBDA. This represents the solution to the problem and a return to the main program is taken.

SUBROUTINE GETLAM - Statements 5 - 10 are executed if BETA = 0. This can occur when NSIG is too large for the machine and EPS computes to zero. In this case, it is easily shown that λ^* is given by the expression in statement 9. At $\beta=0$, all the X_i's become zero except for X_1, which becomes 1. [10] is easily solved for λ. Similarly, since all $X_i = 0$ it is obvious that [11] becomes $+\infty$. Since the search algorithm only depends on the sign of DB and DL and not their magnitude, DB is set to an artificial positive value and a return is taken to MLHE.

Statements 11 - 22 compute the values of the X_i's (vecter X) and the X'_is (vector DX) for the given value of BETA.

Statements 23 - 28 handle the situation where $\beta=1$. In this case, $X_i = N_i$ for all i and λ^* = total failures/total trials is easily shown by setting [10] to zero, setting $X_i = N_i$ and solving for λ.

Statements 29 - 51 perform the same binary search algorithm on λ that is performed on β by MLHE. The loop termination conditions are the same and for the same reasons. Since it can be shown, mathematically, that [10] = $+\infty$ at $\lambda=0$ and [10] = $-\infty$ at $\lambda=\lambda max$, these values are selected as the initial limits on the bracket. Statement 54 calls value to calculate DB using the given BETA and LAMBDA = λ^*.

SUBROUTINE VALUE - The purpose of this routine is to compute the values of the first partials ([10] and [11]) at LAMBDA and BETA. Actually, a scaled value of DB is calculated. This is done to exploit commonality between [10] and [11]. Since the search algorithms depend only on the sign of the derivatives, this scaling does not effect the solution. To understand the scaling, consider [10] and [11] rewritten as follows:

$$\frac{\delta L}{\delta \lambda} = \sum_{i=1}^{K} X_i \left(\frac{F_i}{\lambda X_i} - \frac{S_i}{N-\lambda X_i} \right) \qquad [15]$$

$$\frac{\delta L}{\delta \beta} = \sum_{i=1}^{K} \lambda X'_i \left(\frac{F_i}{\lambda X_i} - \frac{S_i}{N-\lambda X_i} \right) \qquad [16]$$

Note that the terms inside the parenthesis of both [15] and [16] are identical. Both equations can be computed with the same algorithm by just substituting X_i or $\lambda X'_i$ in the first position of each term. Note also that, since $\lambda > 0$, dividing [16] by λ will not effect whether it evaluates to a negative, zero or positive value. This is exploited by VALUE which uses a dummy vector (C) to toggle between X_i and X'_i depending on whether DL or DB is desired. To compute DL, VALUE is called with C = X. To compute the scaled value of DB VALUE is called with C = DX.

SUBROUTINE MESSAGE - This subroutine adds nothing to the solution of λ and β. It is provided for convenience and to explain the meanings of the various values of MSG. It may be freely replaced. It is invoked only from the provided optional main program.

APPENDIX

The next page is blank.

```
      PROGRAM MLHEPRO
C
C  THIS PROGRAM COMPUTES THE MAXIMUM LIKLIHOOD ESTIMATES OF THE
C  PARAMETERS FOR THE AMSAA RELIABILITY GROWTH MODEL AS APPLIED TO
C  DISCRETE DATA.  THE VARIABLES USED ARE DEFINED AS FOLLOWS:
C
C      IOPT-INPUT-AN INTEGER VALUE WHICH SELECTS ONE OF TWO OPTIONS.
C                 IOPT=0 SELECTS THE SPECIAL CASE FOR WHICH ALL
C                 INTERVALS CONTAIN EXACTLY ONE TRIAL.  IOPT=1
C                 SELECTS THE GENERAL CASE FOR WHICH EACH INTERVAL
C                 CAN CONTAIN ANY NUMBER OF TRIALS.
C      K-INPUT-AN INTEGER VALUE WHICH SPECIFIES THE NUMBER OF
C              INTERVALS IN THE CURRENT DATA SET.
C      N-INPUT-A REAL ARRAY WHICH CONTAINS THE NUMBER OF TRIALS
C              IN EACH INTERVAL.  THAT IS, N(I) IS THE NUMBER
C              OF TRIALS IN THE ITH INTERVAL.
C      F-INPUT-A REAL ARRAY WHICH CONTAINS THE NUMBER OF FAILURES
C              IN EACH INTERVAL.  THAT IS, F(I) IS THE NUMBER OF
C              TRIALS IN THE ITH INTERVAL.
C
C  NOTE THAT THE MAXIMUM NUMBER OF INTERVALS ALLOWED IS 500.  TO
C  INCREASE THE NUMBER ALLOWABLE JUST RESET THE DIMENSION LIMITS
C  ON EACH OF THE ARRAYS IN THE DIMENSION STATEMENT.
C
C  THE AMSAA RELIABILITY GROWTH MODEL FOR DESCRETE DATA WAS DEVELOPED
C    BY DR. LARRY CROW OF AMSAA.
C
C  THE COMPUTER PROGRAM TO SOLVE THE MODEL WAS DEVELOPED BY MR. WILLIAM
C    CLAY OF AMSAA.  ANY QUESTIONS CONCERNING THE PROGRAM SHOULD BE
C    DIRECTED TO MR. CLAY AT AV298-6887 OR COMMERCIAL (301) 278-6887
C
C
      DIMENSION N(500),F(500),S(500),X(500),DX(500)
      CHARACTER*1 ANS
      REAL N, LAMBDA
      NSET = 0
    2 PRINT 1
    1 FORMAT(///' ENTER 1 FOR GENERAL CASE MODEL',
     +        /' ENTER 0 FOR ROUND-BY-ROUND MODEL',
     +        /' ENTER ANYTHING ELSE TO STOP',/)
C
C  READ IN THE OPTION AND THE NUMBER OF INTERVALS FOR THE NEXT
C  DATA SET.
C
      READ *,IOPT
      IF (IOPT .LT. 0 .OR. IOPT .GT. 1) STOP
      PRINT 110
C
C NOTE:  THE $ IN THE FORMAT STATEMENT CAUSES THE NEXT READ TO BE
C        SOLICITED ON THE SAME LINE AS THE PRINT.  IF THIS OPTION
C        IS NOT IMPLEMENTED ON YOUR COMPUTER, THE $'S SHOULD BE
C        REMOVED FROM THE FORMAT STATEMENTS.
C
  110 FORMAT(/' ENTER THE NUMBER OF INTERVALS - '$)
```

```
      READ *, K
      IF (K .LT. 2) THEN
        PRINT 120
  120   FORMAT('   MODEL REQUIRES AT LEAST 2 INTERVALS')
        GO TO 2
      ENDIF
      NSET = NSET + 1
      IF (IOPT .EQ. 1) THEN
C
C IF GENERAL CASE OPTION WAS SELECTED (IOPT = 1).  READ IN THE
C NUMBER OF TRIALS AND THE NUMBER OF FAILURES FOR EACH INTERVAL
C IN ORDER.
C
        DO 6 I = 1,K
          PRINT 15, I
   15     FORMAT(' ENTER NO. ROUNDS, FAILURES FOR INTERVAL ',I3,' - '$)
          READ *, N(I), F(I)
          S(I) = N(I) - F(I)
    6   CONTINUE
        PRINT 3,NSET,K
    3   FORMAT(/' DATA SET ',I5,'   NUMBER OF GROUPS IS ',I5,/,
     *  1H ,3('       N       S       F',7X))
        PRINT 4,(N(I),S(I),F(I),I=1,K)
    4   FORMAT(1H ,3F6.0,7X,3F6.0,7X,3F6.0)
      ELSE
C
C FOR THE SPECIAL CASE OPTION (IOPT = 0) READ IN THE NUMBER OF
C FAILURES FOR EACH INTERVAL.  IT IS ASSUMED THAT THE NUMBER OF
C TRIALS FOR EACH INTERVAL IS EXACTLY ONE.
C
        DO 199 I = 1, K
          F(I) = 0.
  199   CONTINUE
  200   PRINT 201
  201   FORMAT(' ENTER SEQUENCE NUMBER OF FAILED ROUND (0 TO QUIT) -'$)
        READ *, INDF
        IF (INDF .LE. 0) THEN
          GO TO 203
        ELSE IF (INDF .LE. K) THEN
          F(INDF) = 1.
        ELSE
          PRINT 202
  202     FORMAT(' INDEX NUMBER EXCEEDS NUMBER OF ROUNDS')
        END IF
        GO TO 200
  203   CONTINUE
        DO 11 I = 1,K
          N(I) = 1.
   11   CONTINUE
        PRINT 12,NSET,K
   12   FORMAT(/' DATA SET ',I5,'  NUMBER OF ROUNDS IS ',I5)
        PRINT 13,(F(I),I=1,K)
   13   FORMAT(20F4.0)
      ENDIF
      CALL MLHE(LAMBDA,BETA,K,N,F,X,DX,4,DB,DL,MSG)
```

```
      IF (MSG .EQ. 0) THEN
        PRINT 5,LAMBDA,BETA
    5   FORMAT(/' ESTIMATED LAMBDA = ',F5.3,'   ESTIMATED BETA = ',F5.3)
        PLAST=1. - LAMBDA * X(K) / N(K)
        PRINT 7,PLAST
    7   FORMAT(' FINAL ESTIMATE OF PROBABILITY OF SUCCESS =  ',F10.8)
      ELSE
        CALL MESSAG(MSG)
      ENDIF
      GO TO 2
      END
```

```
      SUBROUTINE MLHE(LAMBDA,BETA,K,N,F,X,DX,NSIG,DB,DL,MSG)
C
C THIS ROUTINE COMPUTES THE MAXIMUM LIKLIHOOD ESTIMATES OF
C THE TWO PARAMETERS, LAMBDA AND BETA.  VARIABLES ARE AS FOLLOWS:
C
C     LAMBDA-OUTPUT-THE MAXIMUM LIKILIHOOD ESTIMATE OF LAMBDA.
C     BETA-OUTPUT-THE MAXIMUM LIKLIHOOD ESTIMATE OF BETA.
C     K-INPUT-THE NUMBER OF INTERVALS.
C     N-INPUT-AN ARRAY CONTAINING THE NUMBER OF TRIALS FOR EACH INTERVAL
C     F-INPUT-AN ARRAY CONTAINING THE NUMBER OF FAILURES FOR
C             EACH INTERVAL.
C     X,DX-WORKING-ARRAYS OF AT LEAST K WORDS EACH USED AS WORKING
C                  STORAGE WITHIN THE SUBROUTINE.
C     NSIG-INPUT-THE NUMBER OF SIGNIFICANT DIGITS REQUIRED IN
C                THE SOLUTION.  NOTE: COMPUTATIONAL TIME AND TRUNCATION
C                PROBLEMS INCREASE SIGNIFICANTLY IF NSIG IS MUCH GREATER
C                THAN 4.
C     DB-OUTPUT-A SCALED VALUE OF THE PARTIAL OF L WITH RESPECT TO BETA
C               THE VALUE IS NOT CORRECT (UNLESS IT IS ZERO), BUT THE
C               SIGN IS.
C     DL-OUTPUT-THE PARTIAL OF L WITH RESPECT TO LAMBDA.  VALUE AND SIGN
C               ARE CORRECT.
C     MSG-OUTPUT-AN INTEGER VALUE DESCRIBING THE CONDITION OF
C                TERMINATION.  SEE SUBROUTINE MESSAG FOR MORE
C                DETAIL.
C
      REAL LAMBDA,N
      LOGICAL LAST
      DIMENSION N(K),F(K),X(K),DX(K)
      LAMBDA = 0.
      BETA = 0.
      MSG = 0
      EPS = 10. ** (-NSIG) / 2.
      TF = 0.
      TN = 0.
      DO 1 I = 1,K
        IF (N(I) .LE. 0.) MSG = 1
        IF (F(I) .LT. 0.) MSG = 5
        IF (F(I) .GT. N(I)) MSG = 4
        IF (MSG .NE. 0) RETURN
        TF = TF + F(I)
        TN = TN + N(I)
    1 CONTINUE
      IF (TF .EQ. 0. .OR. TN .EQ. TF) THEN
        MSG = 2
        RETURN
      ENDIF
      BETA = 2.
      CALL GETLAM(LAMBDA,BETA,K,N,F,X,DX,EPS,DB,DL,TF,TN)
      IF (DB .GE. 0.) THEN
        MSG = 3
        RETURN
      ENDIF
      BH = 2.
      BL = 0.
```

```
      LAST = .FALSE.
      LOOPS = 0
    2 IF (BH-BL .LE. EPS .OR. LOOPS .GE. 100) LAST = .TRUE.
      LOOPS = LOOPS + 1
      BETA = (BH + BL) / 2.
      CALL GETLAM(LAMBDA,BETA,K,N,F,X,DX,EPS,DB,DL,TF,TN)
      IF(DB.EQ.0.) THEN
        RETURN
      ELSEIF (DB .GT. 0.) THEN
        BL = BETA
      ELSE
        BH = BETA
      ENDIF
      IF (.NOT. LAST) GO TO 2
   10 BETA = (BH + BL) / 2.
      CALL GETLAM(LAMBDA,BETA,K,N,F,X,DX,EPS,DB,DL,TF,TN)
      RETURN
      END
```

```
      SUBROUTINE GETLAM(LAMBDA,BETA,K,N,F,X,DX,EPS,DB,DL,TF,TN)
      REAL LAMBDA,N,LMIN,LMAX
      LOGICAL LAST
      DIMENSION N(K),F(K),X(K),DX(K)
      IF (BETA .EQ. 0.) THEN
        LAMBDA = TF * N(1) / (TF + N(1) - F(1))
        DL = 0.
        DB = 1.
        RETURN
      ENDIF
      SUM = 0.
      PEX = 0.
      PDEX = 0.
      DO 1 I = 1,K
        SUM = SUM + N(I)
        EX = SUM ** BETA
        DEX = EX * ALOG(SUM)
        X(I) = EX - PEX
        DX(I) = DEX - PDEX
        PEX = EX
        PDEX = DEX
    1 CONTINUE
      IF (BETA .EQ. 1.) THEN
        LAMBDA = TF / TN
        DL = 0.
        CALL VALUE(LAMBDA,K,N,F,X,DX,DB)
        RETURN
      ENDIF
      LMIN = 0.
      LMAX = N(1) / X(1)
      IF (BETA .GT. 1.) LMAX = N(K) / X(K)
      UPRLIM = LMAX
      LAST = .FALSE.
      LOOPS = 0
    2 IF (LMAX - LMIN .LE. EPS .OR. LOOPS .GE. 100) LAST = .TRUE.
      LOOPS = LOOPS + 1
      LAMBDA = (LMAX + LMIN) / 2.
      CALL VALUE(LAMBDA,K,N,F,X,X,DL)
      IF(DL.EQ.0.) THEN
        GO TO 100
      ELSEIF (DL .GT. 0.) THEN
        LMIN = LAMBDA
      ELSE
        LMAX = LAMBDA
      ENDIF
      IF(.NOT. LAST) GO TO 2
      IF (LMAX .NE. UPRLIM .AND. LMIN .NE. 0.) THEN
        LAMBDA =  (LMIN + LMAX) / 2.
        CALL VALUE(LAMBDA,K,N,F,X,X,DL)
      ENDIF
  100 CALL VALUE(LAMBDA,K,N,F,X,DX,DB)
      RETURN
      END
```

```
      SUBROUTINE VALUE(LAMBDA,K,N,F,X,C,DIRIV)
      REAL LAMBDA,N
      DIMENSION N(K),F(K),X(K),C(K)
      DIRIV = 0.
      DO 100 I = 1,K
        EX = LAMBDA * X(I)
        DIRIV = DIRIV + C(I) * N(I) * (F(I) - EX) / ((N(I) - EX) * EX)
  100 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE MESSAG(M)
      IF(M.EQ.1) PRINT 1
    1 FORMAT(' AN INTERVAL WAS DETECTED WITH LESS THAN 1 TRIAL',
     +/' NO SOLUTION WAS ATTEMPTED')
      IF(M.EQ.2) PRINT 2
    2 FORMAT(' TOTAL NUMBER OF FAILURES IS ZERO OR TOTAL NUMBER',
     +/' OF SUCCESSES IS ZERO.  NO SOLUTION WAS ATTEMPTED')
      IF(M.EQ.3) PRINT 3
    3 FORMAT(' THE ESTIMATE OF BETA EXCEEDS 2',
     +/' NO SOLUTION WAS ATTEMPTED')
      IF(M.EQ.4) PRINT 4
    4 FORMAT(' AN INTERVAL WAS DETECTED WITH MORE FAILURES THAN ',
     +'TRIALS'/' NO SOLUTION WAS ATTEMPTED')
      IF(M.EQ.5) PRINT 5
    5 FORMAT(' AN INTERVAL WAS DETECTED WITH LESS THAN ZERO ',
     +'FAILURES'/' NO SOLUTION WAS ATTEMPTED')
      RETURN
      END
```

DISTRIBUTION LIST

No. of Copies	Organization
12	Commander Defense Technical Information Center ATTN: DTIC-DDAC Cameron Station, Bldg 5 Alexandria, VA 22314-6145
3	Commander US Army Materiel Command ATTN: AMCDMA-M AMCDMA-MS (2 cys) 5001 Eisenhower Avenue Alexandria, VA 22333-0001
2	Director US Army TRADOC Systems Analysis Activity ATTN: ATOR-TSL ATOR-T White Sands Missile Range, NM 88002-5502
1	Commander US Army Concepts Analysis Agency ATTN: CSCA-MSI-L 8120 Woodmont Avenue Bethesda, MD 20814-2797
1	Director Combat Data Information Center AFWAL/FIES/CDIC Wright-Patterson AFB, OH 45433-5000
1	Pentagon Library ATTN: ANR-AL-RS (Army Studies) Pentagon, Rm 1A518 Washington, DC 20310
3	Chief Defense Logistics Studies Information Exchange US Army Logistics Management Center ATTN: AMXCM-D (2 cys) AMXCM-ACM (Mr. Fowler) Fort Lee, VA 23801-6043
2	Commander US Army Development and Employment Agency ATTN: MODE-TED-SAB Ft. Lewis, WA 98433-5000
1	Commandant US Army Infantry School ATTN: ATSH-CD-CS-OR Fort Benning, GA 31905-5400
1	Commander USAREUR & 7th Army ATTN: AEAG-X-OR APO New York 09403

Aberdeen Proving Ground

No. of Copies	Organization
1	Dir, BRL, Bldg 328 ATTN: SLCBR-D APG, MD 21005-5066
1	Dir, BRL ATTN: SLCBR-OD-ST Bldg 305 APG, MD 21005-5066

END
Feb.
1988
DTIC